Bankruptcy Prediction: Mining the Polish Bankruptcy Data

Sai Surya Teja Maddikonda
maddikonda.s@husky.neu.edu

Sree Keerthi Matta
matta.s@husky.neu.edu

Abstract

Bankruptcy prediction is the task of predicting bankruptcy and various measures of financial distress of firms. It is a vast area of finance and accounting research. The importance of the area is due in part to the relevance for creditors and investors in evaluating the likelihood that a firm may go bankrupt. The aim of predicting financial distress is to develop a predictive model that combines various econometric parameters which allow foreseeing the financial condition of a firm. In this domain various methods were proposed that were based on statistical hypothesis testing, statistical modeling (e.g., generalized linear models), and recently artificial intelligence (e.g., neural networks, Support Vector Machines, decision trees). In this paper we document our observations as we explore, build, and compare, some of the widely used classification models: Extreme Gradient Boosting for Decision Trees, Random Forests, Naïve Bayes, Balanced Bagging and Logistic Regression, pertinent to bankruptcy prediction. We have chosen the Polish companies' bankruptcy data set where synthetic features were used to reflect higher-order statistics. A synthetic feature is a combination of the econometric measures using arithmetic operations (addition, subtraction, multiplication, division). We begin by carrying out data preprocessing and exploratory analysis where we impute the missing data values using some of the popular data imputation techniques like Mean, k-Nearest Neighbors, Expectation-Maximization and Multivariate Imputation by Chained Equations (MICE). To address the data imbalance issue, we apply Synthetic Minority Oversampling Technique (SMOTE) to oversample the minority class labels. Later, we model the data using K-Fold Cross Validation on the said models, and the imputed and resampled datasets. Finally, we analyze and evaluate the performance of the models on the validation datasets using several metrics such as accuracy, precision, recall, etc., and rank the models accordingly. Towards the end, we discuss the challenges we faced and suggest ways to improve the prediction, including scope for future work.

1. Introduction

Prediction of an enterprise bankruptcy is of great importance in economic decision making. A business condition of either small or large firm concerns local community, industry participants and investors, but also influences policy makers and global economy. Therefore, the high social and economic costs because of corporate bankruptcies have attracted attention of researchers for better understanding of bankruptcy causes and eventually prediction of business distress [2]. The quantity of research in this area is also a function of the availability of data: for public firms which went bankrupt or did not, numerous accounting ratios that might indicate danger can be calculated, and numerous other potential explanatory variables are also available. Consequently, the area is well-suited for testing of increasingly sophisticated, data-intensive forecasting approaches [3].

The history of bankruptcy prediction includes application of numerous statistical tools which gradually became available, and involves deepening appreciation of various pitfalls in early analyses. Interestingly, research is still published that suffers pitfalls that have been understood for many years. Bankruptcy prediction has been a subject of formal analysis since at least 1932, when FitzPatrick published a study of 20 pairs of firms, one failed and one surviving, matched by date, size and industry, in The Certified Public Accountant. He did not perform statistical analysis as is now common, but he thoughtfully interpreted the ratios and trends in the ratios. His interpretation was effectively a complex, multiple variable analysis.

The purpose of the bankruptcy prediction is to assess the financial condition of a company and its future perspectives within the context of long-term operation on the market [4]. It is a vast area of finance and econometrics that combines expert knowledge about the phenomenon and historical data of prosperous and unsuccessful companies. Typically, enterprises are quantified by numerous indicators that describe their business condition that are further used to induce a mathematical model using past observations [5].

There are different issues that are associated with the bankruptcy prediction. Two main problems are the following: First, the econometric indicators describing the firm's condition are pro- posed by domain experts. However, it is rather unclear how to combine them into a successful model. Second, the historical observations used to train a model are usually influenced by imbalanced data phenomenon, because there are typically much more successful companies than the bankrupted ones. As a consequent, the trained model tends to predict companies as successful (majority class) even when some of them are

distressed firms. Both these issues mostly influence the final predictive capability of the model.

To speak about the modern methods of approach for the field of bankruptcy prediction, it is worth noting that survival methods are now being applied. Option valuation approaches involving stock price variability have been developed. Under structural models, a default event is deemed to occur for a firm when its assets reach a sufficiently low level compared to its liabilities. Neural network models and other sophisticated models have been tested on bankruptcy prediction. Modern methods applied by business information companies surpass the annual accounts content and consider current events like age, judgements, bad press, payment incidents and payment experiences from creditors.

1.1 Previous work

First attempts of the formal bankruptcy prediction trace back to the beginnings of the 20th century when first econometric indicators were proposed to describe predictive abilities of business failure (Fitzpatrick, 1932; Merwin, 1942; Winakor & Smith, 1935) [6]. The sixties of the twentieth century brought a turning point in the survey of the early recognition of the business failure symptoms. First of all, the work of Beaver (1966) initiated application of statistical models to the bankruptcy prediction. Following this line of thinking, Altman (1968) proposed to use multidimensional analysis to predict corporate bankruptcy that was further developed by others (Altman & Loris, 1976; Blum, 1974; Deakin, 1972; Edmister, 1972; Ketz, 1978; Koh & Killough, 1990; Laitinen, 1991; Libby, 1975; Meyer & Pifer, 1970; Pettway & Sinkey, 1980; Rujoub, Cook, & Hay, 1995; Sinkey, 1975; Wilcox, 1973) [7]. In parallel, a great interest was paid to the generalized linear models that can be used in both decision making and providing certainty of the prediction (Aziz, Emanuel, & Lawson, 1988; Grice & Dugan, 2003; Hopwood, McKeown, & Mutchler, 1994; Koh, 1991; Li & Miu, 2010; Ohlson, 1980; Platt & Platt, 1990; Platt, Platt, & Pedersen, 1994; Zavgren, 1983; Zmijewski, 1984) [8]. Additionally, the generalized linear models are of special interest because estimated weights of the linear combination of economic indicators in the model can be further used to determine importance of the economic indicators.

Since nineties of the 20th century artificial intelligence and machine learning have become a major research direction in the bankruptcy prediction. In the era of increasing volumes of data, it turned out that the linear models like the logistic regression or logit (probit) models are unable to reflect non-trivial relationships among economic metrics. Moreover,

the estimated weights of the linear models are rather unreliable to indicate the importance of the metrics.
In order to obtain comprehensible models with an easy to understand knowledge representation, decision rules expressed in terms of first-order logic were induced using different techniques, naming only a few, like rough sets (Dimitras, Slowinski, Susmaga, & Zopounidis, 1999)[9] or evolutionary programming (Zhang et al., 2013). However, the classification accuracy of the decision rules are very often insufficient, therefore, more accurate methods were applied to the bankruptcy prediction. One of the most successful model was support vector machines (SVM) (Shin, Lee, & Kim, 2005). The disadvantages of SVM are that the kernel function must be carefully hand-tuned and it is impossible to obtain comprehensible model.

A different approach aims at automatic feature extraction from data, i.e., automatic non-linear combination of econometric indicators, which alleviates the problem of a specific kernel function determination in the case of SVM. This approach applies neural networks to the bankruptcy prediction (Bell, Ribar, & Verchio, 1990; Cadden, 1991; Coats & Fant, 1991; Geng, Bose, & Chen, 2015; Koster, Sondak, & Bourbia, 1991; Salchenberger, Cinar, & Lash, 1992; Serrano-Cinca, 1996; Tam, 1991; Tam & Kiang, 1992; Wilson & Sharda, 1994; Zhang, Hu, Patuwo, & Indro, 1999) [10]. The main problem of the neural networks lies in the fact that they can fail in case of multimodal data. Typically, the econometric metrics need to be normalized/standardized in order to have all features of the same magnitude. This is also necessary for training neural networks so that the errors could be backpropagated properly. However, the normalization/standardization of data do not reduce the problem of the data multimodality that may drastically reduce predictive capabilities of the neural networks. That is why it has been advocated to take advantage of different learning paradigm, namely, the ensemble of classifiers (Kittler, Hatef, Duin, & Matas, 1998) [11]. The idea of the ensemble learning is to train and combine typi- cally weak classifiers to obtain better predictive performance. First approaches but still very successful were bagging (Breiman, 1996)[12] and boosting (Freund & Schapire, 1996; Friedman, 20 01; 20 02; Zi¸eba, Tomczak, Lubicz, & ´Swi¸atek, 2014)[13]. The idea of boosting was further developed to the case of unequal classification costs (Fan, Stolfo, Zhang, & Chan, 1999) and imbalanced data (Galar, Fernandez, Barrenechea, Bustince, & Herrera, 2012)[14]. Recently, the boosting method was modified to optimize a Taylor expansion of the loss functions, an approach known as Extreme Gradient Boosting (Chen & He, 2015a) that obtains state-of-the-art results in many problems on Kaggle competitions. 1 Recently, it has been shown that the ensemble classifier can be successfully applied to the bankruptcy prediction (Nanni & Lumini, 2009)[15] and it significantly beats other methods (Alfaro, García, Gámez, & Elizondo, 2008)[16].

1.2 Recent work

The latest research within the field of Bankruptcy and Insolvency Prediction compares various differing approaches, modelling techniques, and individual models to ascertain whether any one technique is superior to its counterparts. Jackson and Wood (2013) provides an excellent discussion of the literature to date, including an empirical evaluation of 15 popular models from the existing literature. These models range from the univariate models of Beaver through the multidimensional models of Altman and Ohlson, and continuing to more recent techniques which include option valuation approaches. They find that models based on market data—such as an option valuation approach—outperform those earlier models which rely heavily on accounting numbers. Zhang, Wang, and Ji (2013)[17] proposed a novel rule-based system to solve bankruptcy prediction problem. The whole procedure consists of the following four stages: first, sequential forward selection was used to extract the most important features; second, a rule-based model was chosen to fit the given dataset since it can present physical meaning; third, a genetic ant colony algorithm (GACA) was introduced; the fitness scaling strategy and the chaotic operator were incorporated with GACA, forming a new algorithm—fitness-scaling chaotic GACA (FSCGACA), which was used to seek the optimal parameters of the rule-based model; and finally, the stratified K-fold cross-validation technique was used to enhance the generalization of the model.

2. Methodology

In the previous section, we formally introduced the problem statement of bankruptcy prediction. In this section, we explain our step-by-step solution of how we achieved benchmark results for bankruptcy prediction. Firstly, we introduce the Polish bankruptcy dataset and explain the details of the dataset like features, instances, data organization, etc. Next, we delve into data preprocessing steps, where we state the problems present with the data like missing data and data imbalance, and explain how we dealt with them. Next, we introduce the classification models we have considered and explain how we train our data using these models. Later, we analyze and evaluate the performance of these models using certain metrics like accuracy, precision and recall.

2.1 Data

The dataset we have considered for addressing the bankruptcy prediction problem is the Polish bankruptcy data, hosted by the University of California Irvine (UCI) Machine Learning Repository—a huge repository of freely accessible datasets for research and learning purposes intended for the Machine Learning/Data Science community. The dataset is about bankruptcy prediction of Polish companies. The data was collected from

Emerging Markets Information Service (EMIS), which is a database containing information on emerging markets around the world. The bankrupt companies were analyzed in the period 2000-2012, while the still operating companies were evaluated from 2007 to 2013. The dataset is very apt for our research about bankruptcy prediction because it has highly useful econometric indicators as attributes (features) and comes with a huge number of samples of Polish companies that were analyzed in 5 different timeframes: Based on the collected data five classification cases were distinguished, that depends on the forecasting period:

1. **1st year**: The data contains financial rates from 1st year of the forecasting period and corresponding class label that indicates bankruptcy status after 5 years.
2. **2nd year**: The data contains financial rates from 2nd year of the forecasting period and corresponding class label that indicates bankruptcy status after 4 years.
3. **3rd year**: The data contains financial rates from 3rd year of the forecasting period and corresponding class label that indicates bankruptcy status after 3 years.
4. **4th year**: The data contains financial rates from 4th year of the forecasting period and corresponding class label that indicates bankruptcy status after 2 years.
5. **5th year**: The data contains financial rates from 5th year of the forecasting period and corresponding class label that indicates bankruptcy status after 1 years.

The dataset is summarized in Table 1 below.

Dataset characteristic	Multivariate			
Number of Features	64			
Number of Instances	**Data**	**Total Instances**	**Bankrupt instances**	**Non-bankrupt instances**
	1st year	7027	271	6756
	2nd year	10173	400	9773
	3rd year	10503	495	10008
	4th year	9792	515	9227
	5th year	5910	410	5500
Feature characteristics	Real values			
Has missing data?	Yes			
Associated tasks	Classification			
Date donated	04-11-2016			

Table 1: Summary of the Polish bankruptcy dataset.

ID	Description	ID	Description
X1	net profit / total assets	**X33**	operating expenses / short-term liabilities
X2	total liabilities / total assets	**X34**	operating expenses / total liabilities
X3	working capital / total assets	**X35**	profit on sales / total assets
X4	current assets / short-term liabilities	**X36**	total sales / total assets
X5	[(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365	**X37**	(current assets - inventories) / long-term liabilities
X6	retained earnings / total assets	**X38**	constant capital / total assets
X7	EBIT / total assets	**X39**	profit on sales / sales
X8	book value of equity / total liabilities	**X40**	(current assets - inventory - receivables) / short-term liabilities
X9	sales / total assets	**X41**	total liabilities / ((profit on operating activities + depreciation) * (12/365))
X10	equity / total assets	**X42**	profit on operating activities / sales
X11	(gross profit + extraordinary items + financial expenses) / total assets	**X43**	rotation receivables + inventory turnover in days
X12	gross profit / short-term liabilities	**X44**	(receivables * 365) / sales
X13	(gross profit + depreciation) / sales	**X45**	net profit / inventory
X14	(gross profit + interest) / total assets	**X46**	(current assets - inventory) / short-term liabilities
X15	(total liabilities * 365) / (gross profit + depreciation)	**X47**	(inventory * 365) / cost of products sold
X16	(gross profit + depreciation) / total liabilities	**X48**	EBITDA (profit on operating activities - depreciation) / total assets
X17	total assets / total liabilities	**X49**	EBITDA (profit on operating activities - depreciation) / sales
X18	gross profit / total assets	**X50**	current assets / total liabilities
X19	gross profit / sales	**X51**	short-term liabilities / total assets
X20	(inventory * 365) / sales	**X52**	(short-term liabilities * 365) / cost of products sold)
X21	sales (n) / sales (n-1)	**X53**	equity / fixed assets
X22	profit on operating activities / total assets	**X54**	constant capital / fixed assets
X23	net profit / sales	**X55**	working capital
X24	gross profit (in 3 years) / total assets	**X56**	(sales - cost of products sold) / sales
X25	(equity - share capital) / total assets	**X57**	(current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X26	(net profit + depreciation) / total liabilities	**X58**	total costs /total sales
X27	profit on operating activities / financial expenses	**X59**	long-term liabilities / equity
X28	working capital / fixed assets	**X60**	sales / inventory
X29	logarithm of total assets	**X61**	sales / receivables
X30	(total liabilities - cash) / sales	**X62**	(short-term liabilities *365) / sales
X31	(gross profit + interest) / sales	**X63**	sales / short-term liabilities
X32	(current liabilities * 365) / cost of products sold	**X64**	sales / fixed assets

Table 2: Summary of feature in the Polish bankruptcy data

Table 1 shows the total number of features and instances in the dataset, and the number of samples in each class (bankrupt or not-bankrupt) of all the 5 datasets. The features are explained in Table 2 above. As shown in the table, there are 64 features labelled X1 through X64, and each feature is a synthetic feature. A synthetic feature is a combination of the econometric measures using arithmetic operations (addition, subtraction, multiplication, division). Each synthetic feature is as a single regression model that is developed in an evolutionary manner. The purpose of the synthetic features is to combine the econometric indicators proposed by the domain experts into complex features. The synthetic features can be seen as hidden features extracted by the neural networks but the fashion they are extracted is different.

2.2 Dataset Quality Assessment

Now we move on to assessing the quality of the dataset. As we have mentioned earlier, the dataset suffers from missing values and data imbalance.

2.2.1 Missing Data

First, we look at some statistics of missing values. As an example, we plot of the nullity matrix for the 1st year dataset that explains the sparsity of 1st Year data. This plot shown in Figure 1 was achieved using the library **`missingno`**. The nullity matrix gives us a data-dense display which lets us visually pick out the missing data patterns in the dataset. We notice that the features **X21** and **X37** have the highest number of missing values.

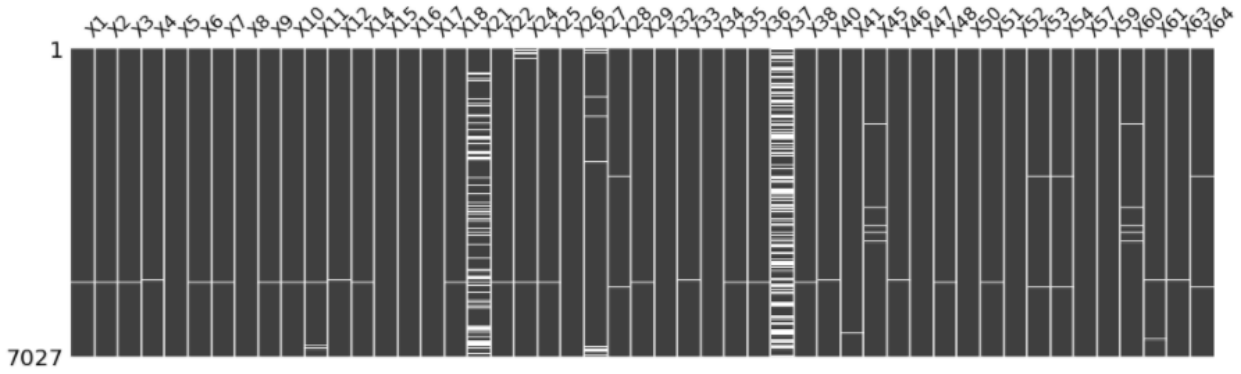

Figure 1: Sparsity matrix for dataset 'Year 1'. The white spaces indicate missing data values for the feature in the corresponding column.

Now, we explore to see the correlation among various features in the 1st Year data as an example. Shown in Figure 2 is a correlation heatmap for the 1st Year data that describes the degree of nullity relationship between the different features. The range of this nullity correlation is from -1 to 1 ($-1 \leq R \leq 1$). Features with no missing value are excluded in the heatmap. If the nullity correlation is very close to zero ($-0.05 < R < 0.05$), no value

will be displayed. Also, a perfect positive nullity correlation (R=1) indicates when the first feature and the second feature both have corresponding missing values while a perfect negative nullity correlation (R=-1) means that one of the features is missing and the second is not missing.

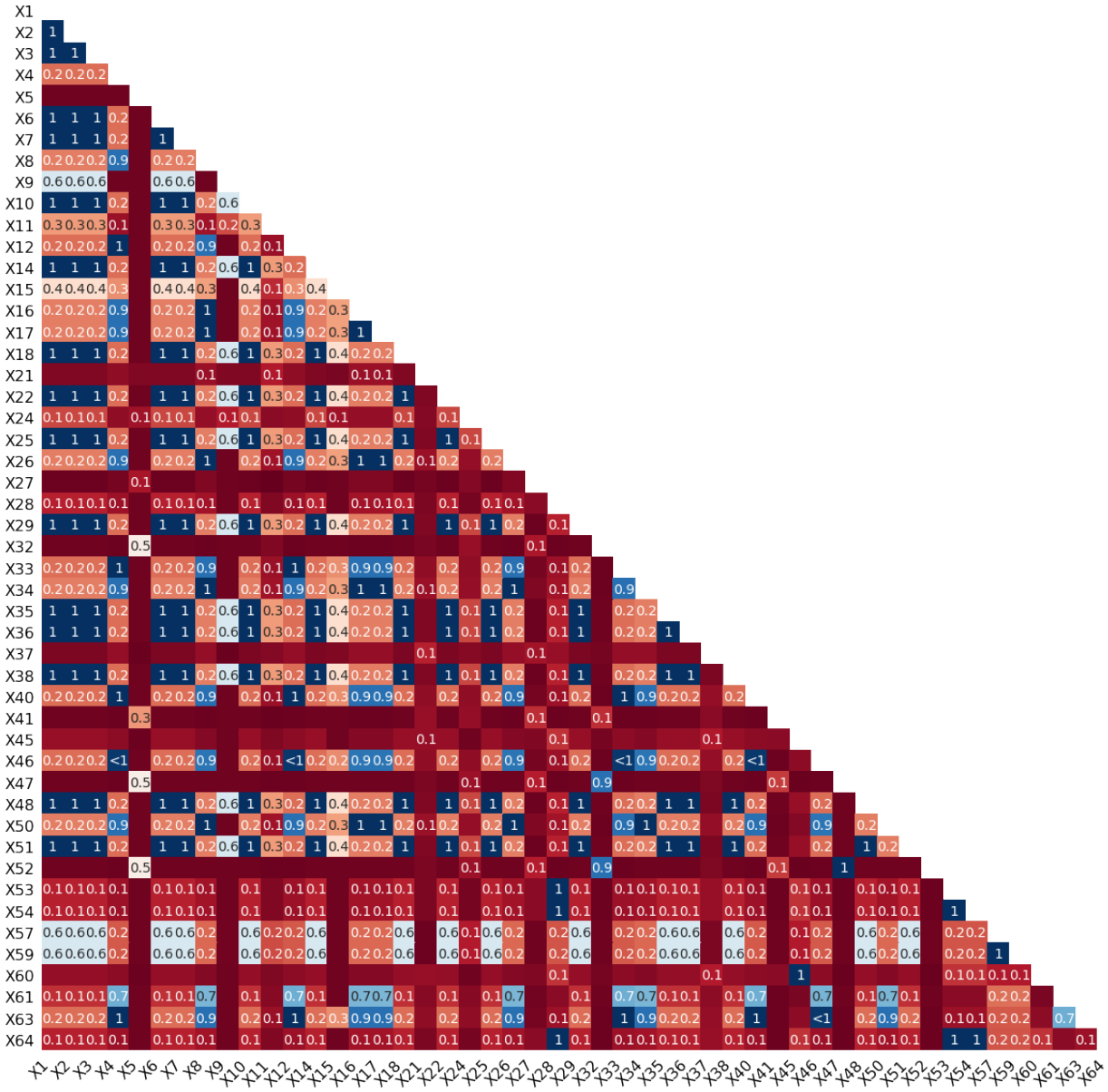

Figure 2: Feature correlation heatmap for the 1st Year dataset.

We have visually seen the sparsity in the data, as well as correlation among the features with respect to missing values. Now, let us see how much of data is actually missing. In Table 3 shown below, the second column shows the total number of instances in each dataset, and third column shows the number of instances or rows with missing values for at least one of the features. A naive approach of dealing with missing values would be to

drop all such rows as in Listwise deletion. But dropping all such rows leads to a tremendous data loss. Column 4 shows the number of instances that would remain in each dataset if all rows with missing values were dropped. Column 5 shows the percent of data loss if all the rows with missing data values were indeed dropped. As the data loss in most of the datasets is over 50%, it is now clear that we cannot simply drop the rows with missing values, as it leads to severe loss in the representativeness of data.

Data Set	**#Total Instances**	**# Instances with missing values**	**# Instances that would remain if all rows with missing values were dropped**	**% Data loss if rows with missing values were dropped**
Year 1	7027	3833	3194	54.54 %
Year 2	10173	6085	4088	59.81 %
Year 3	10503	5618	4885	53.48 %
Year 4	9792	5023	4769	51.29 %
Year 5	5910	2879	3031	48.71 %

Table 3: Assessing the Missing Data for all the datasets.

2.2.2 Data Imbalance

We have covered the missing data aspect of the data quality assessment, let us now see the Data Imbalance aspect. Table 4 shown below summarizes the populations of class labels in each dataset. Column 2 shows the total instances, while Column 3 and Column 4 show the number of instances with class label as Bankrupt and Non-Bankrupt respectively. Looking at the numbers of Bankrupt class label, we can figure out that they are a minority when compare with the non-bankrupt class label. But Column 5 clearly shows the population percentage of the minority class, i.e., the Bankruptcy class label, among the total population of the dataset. These numbers in column 5 tell us that there is a huge data imbalance. If this imbalance is not cured, in the modeling stage that follows, the models will not have seen enough data from the minority class label and they train and hence perform poorly.

2.3 Dealing with Missing Data

Missing data causes 3 problems:

1. Missing data can introduce a substantial amount of bias.
2. Makes the handling and analysis of the data more difficult.
3. Create reductions in efficiency.

Data Set	# Total Instances	# Bankrupt instances in this forecasting period	# Non-Bankrupt instances in this forecasting period	Percentage of minority class samples
Year 1	7027	271	6756	3.85 %
Year 2	10173	400	9773	3.93 %
Year 3	10503	495	10008	4.71 %
Year 4	9792	515	9277	5.25 %
Year 5	5910	410	5500	6.93 %

Table 4: Assessing the Data Imbalance for all the datasets.

Dropping all the rows with missing values or Listwise deletion, introduces bias and affects representativeness of the results. The only viable alternative to Listwise deletion of missing data is Imputation. Imputation is the process of replacing missing data with substituted values and it preserves all the cases by replacing missing data with an estimated value, based on other available information. In our project we explored 4 techniques of imputation, and we will see them in the subsequent sections.

1. Mean Imputation
2. k-Nearest Neighbors Imputation
3. Expectation-Maximization Imputation
4. Multivariate Imputation Using Chained Equations

2.3.1 Mean Imputation

Mean imputation technique is the process of replacing any missing value in the data with the mean of that variable in context. In our dataset, we replaced a missing value of a feature, with the mean of the other non-missing values of that feature. Mean imputation attenuates any correlations involving the variable(s) that are imputed. This is because, in cases with imputation, there is guaranteed to be no relationship between the imputed variable and any other measured variables. Thus, mean imputation has some attractive properties for univariate analysis but becomes problematic for multivariate analysis. Hence we opted Mean Imputation as a baseline method. We achieved mean imputation using scikit-learn's **`Imputer`** class.

2.3.2 k-Nearest Neighbors Imputation

The k-nearest neighbors algorithm or k-NN, is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. It can also be used as a data imputation technique k-NN

imputation replaces NaNs in Data with the corresponding value from the nearest-neighbor row or column depending upon the requirement. The nearest-neighbor row or column is the closest row or column by Euclidean distance. If the corresponding value from the nearest-neighbor is also NaN, the next nearest neighbor is used. We used the **`fancyimpute`** library to perform k-NN data imputation, and we used 100 nearest neighbors for the process.

2.3.3 Expectation-Maximization Imputation

In statistics, an EM or expectation–maximization algorithm is an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters. It is followed by a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step. EM Imputation is the process of imputing missing values using Expectation-Maximization. Missing values of quantitative variables are replaced by their expected value computed using the Expectation-Maximization (EM) algorithm. In practice, a Multivariate Gaussian distribution is assumed. In general, EM imputation is better than mean imputations because they preserve the relationship with other variables. We achieved EM Imputation using **`impyute`** library.

2.3.4 Multivariate Imputation using Chained Equations

Multiple imputation using chained equations or MICE is an imputation technique that uses multiple imputations as opposed to single imputation. MICE is regarded as a fully conditional specification or sequential regression multiple imputation. It has become one of the principal methods of addressing missing data. Creating multiple imputations, as opposed to single imputations, accounts for the statistical uncertainty in the imputations. In addition, the chained equations approach is very flexible and can handle variables of varying types (for example., continuous or binary), as well as complexities such as bounds or survey skip patterns. MICE is beneficial when the missing data is large. Because multiple imputation involves creating multiple predictions for each missing value, the analyses of multiply imputed data take into account the uncertainty in the imputations and yield accurate standard errors. In the MICE procedure a series of regression models are run whereby each variable with missing data is modeled conditional upon the other variables in the data. This means that each variable can be modeled according to its distribution, with, for example, binary variables modeled using logistic regression and

continuous variables modeled using linear regression. We achieved MICE imputation using **`fancyimpute`** library.

2.3 Dealing with Data Imbalance

Moving on to the other shortcoming of the Polish bankruptcy dataset, we now explain how we dealt with the Data Imbalance. Data Imbalance can be treated with Oversampling and/or Undersampling. In data analysis, Oversampling and Undersampling are opposite and roughly equivalent techniques of dealing with Data Imbalance, where they adjust the class distribution of a data set (i.e. the ratio between the different classes/categories represented). Oversampling is increasing the class distribution of the minority class label whereas Undersampling is decreasing the class distribution of the majority class label. In our project, we explored Synthetic Minority Oversampling Technique or SMOTE.

2.3.1 Synthetic Minority Oversampling Technique (SMOTE)

Synthetic Minority Oversampling Technique (SMOTE) is a widely used oversampling technique. To illustrate how this technique works consider some training data which has s samples, and f features in the feature space of the data. For simplicity, assume the features are continuous. As an example, let us consider a dataset of birds for clarity. The feature space for the minority class for which we want to oversample could be beak length, wingspan, and weight. To oversample, take a sample from the dataset, and consider its k nearest neighbors in the feature space. To create a synthetic data point, take the vector between one of those k neighbors, and the current data point. Multiply this vector by a random number x which lies between 0, and 1. Adding this to the current data point will create the new synthetic data point. SMOTE was implemented from the **`imbalanced-learn`** library.

2.4 Data Modeling

In this section, we will look at the various classification models that we have considered for training on the Polish bankruptcy datasets to achieve the task of coming up with a predictive model that would predict the bankruptcy status of a given (unseen) company with an appreciable accuracy. We have considered the following 6 models:

1. Gaussian Naïve Bayes
2. Logistic Regression
3. Decision Tree
4. Random Forests
5. Extreme Gradient Boosting
6. Balanced Bagging

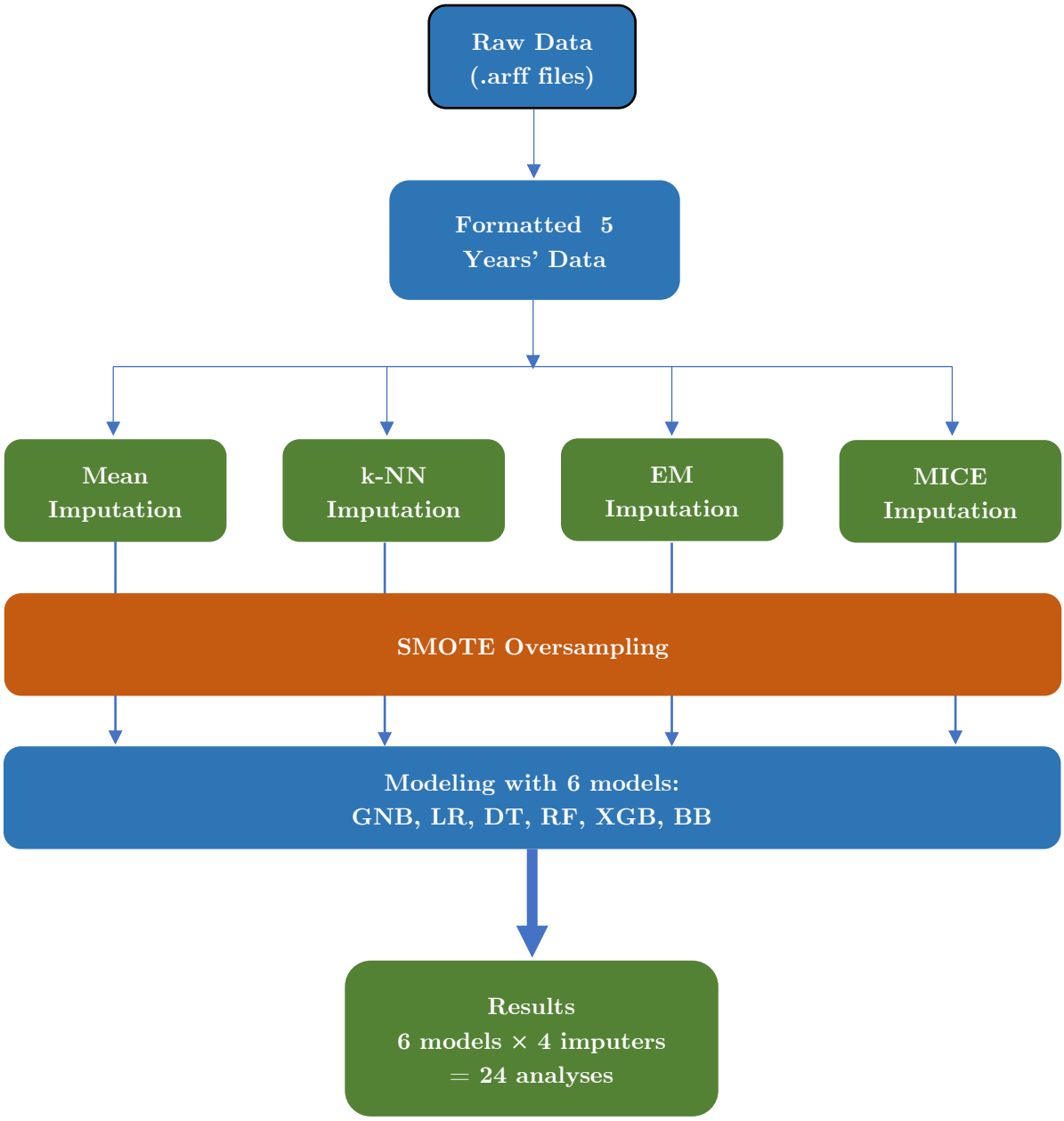

Figure 3: Pipeline for data modeling

Figure 3 shows the pipeline of data modeling for our project. After having obtained the formatted datasets from the raw data (.arff files), we have imputed the missing values via 4 different independent imputer methods (mean, k-NN, EM and MICE). Later, we have oversampled all these 4 imputed datasets with SMOTE oversampling technique and thus obtained 4 imputed and oversampled datasets. They datasets are ready for the data

modeling step. We model each of these 4 datasets with the 6 models listed above. While modeling, we use the K-Fold Cross Validation technique for validation.

K-Fold Cross Validation

Since the Polish bankruptcy dataset does not have a separate 'unlabeled' test dataset, it is obvious that we need to split the training data to obtain a validation dataset (for each year's data). If the split was done in a simple way, we end up with just one validation dataset and the inherent difference in the class label distributions for training and validation datasets would lead to poor performance of the model on the training and hence on validation sets. Alternatively, in K-Fold Cross Validation, the training dataset is split into K bins. In each iteration (total = K iterations), one bin is retained as a validation dataset and the other bins of data are used for training the model. The performance metrics (like accuracy, precision, recall, etc) are noted for each validation set. After all the iterations, each of the bins will have served as validation dataset at least once (depending on K). The metrics are averaged over all the K iterations and the final metrics are output.

Hence, towards the end of the modeling step, we obtain 24 different results (6 models × 4 imputer datasets). In each of the sub-sections that follow, we first explain the model briefly, explain the experiment by specifying the hyperparameters of the model. Later, in the Results section we report the (Cross-Validation-Average) performance of the model on the validation data.

2.4.1 Gaussian Naïve Bayes Classifier

Naive Bayes classifier is one of the supervised learning algorithms which is based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \ \ldots, \ x_n) = \frac{P(y)P(x_1, \ \ldots, \ x_n \mid y)}{P(x_1, \ \ldots, \ x_n)}$$

Using the naive independence assumption that:

$$P(x_i \mid y, \ x_1, \ \ldots, \ x_{i-1}, \ x_{i+1}, \ \ldots x_n) = P(x_i \mid y)$$

for all i, this relationship is simplified to:

$$P(y \mid x_1, \ \ldots, \ x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ \ldots, \ x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ \ldots, \ x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Gaussian Naïve Bayes implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

2.4.2 Logistic Regression Classifier

Logistic regression is a linear model for classification. It is also known as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

$$\text{LogitClassifier}(x) = \min_{w,c} ||w||_1 + C \sum_{i=1}^{n} \left(\log(\exp\left(-y\,(X_i^T w + c)\right) + 1)\right)$$

Introduced L1 penalty (Lasso) into the model, which calculates LogitClassifier value for data points x_i as follows:

$$\text{LogitClassifier}(x) = \text{LogitClassifier}^*(x) - \lambda J(\theta)$$

$$J(\theta) = \sum_{i=1}^{n} ||x_i||$$

We implemented the model with $\lambda = 1$ and equal weights are given for all the features, using L1 regularization.

2.4.3 Decision Trees Classifier

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. For our classification task, we create a model that predicts the value of a target variable (y = will a firm go bankrupt?) by learning simple decision rules inferred from the data features ($x_1, x_2 \ldots x_{64}$ - all the financial distress variables of a firm). While building decision tree, the data comes in records in the form:

$$(x, Y) = (x_1, x_2, \ldots\ldots, x_{64}, Y)$$

Our model considers all features and gives equal weights to each of them while looking for best split during construction of a decision tree. We have considered 'Gini' index as a measure the quality of a split.

2.4.4 Random Forests Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy

and control over-fitting. In random forests, each tree in the ensemble is built from a sample drawn with replacement from the training set. Also, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model. In our model, the number of estimators used are 5 and we have considered 'Entropy' as a measure of the quality of a split.

2.4.5 Extreme Gradient Boosting Classifier

Extreme Gradient Boosting (XGBoost) is built on the principles of gradient boosting framework. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. XGBoost uses a more regularized model formalization to control over-fitting, which gives it better performance. In our model, the number of estimators used are 100. The model internally uses log-linear classifier for regularizing the model with $\lambda = 1$.

2.4.6 Balanced Bagging Classifier

Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees used for classification task. We know that decision trees are sensitive to the specific data on which they are trained. If the training data is changed the resulting decision tree can be quite different and in turn the predictions can be quite different. Balancing the data set before training the classifier improve the classification performance. In addition, it avoids the ensemble to focus on the majority class which would be a known drawback of the decision tree classifiers. We have used base estimator as Random Forests in order to perform Balanced Bagging. The number of estimators used in our model are 5. We have considered 'Entropy' as a measure of the quality of a split.

3. Code

The programming environment used for the project is Python v3.6. We used an Intel Core i5 2.5 GHz Dual Core processor with 8 GB Memory (RAM) and 1 TB of storage (disk space) to run our experiments. Our code workflow exactly mimics the data modeling pipeline shown in Figure 3.

We used the libraries listed in Table 5 to run our experiments and achieve our results.

Library	Description
numpy	Data organization and statistical operations.
pandas	Data manipulation and analysis. Storing and manipulating numerical tables.
matplotlib	Plotting library
scipy.io	Loading .arff raw data
missingno	Generate nullity matrices and correlation heatmaps for missing data.
fancyimpute	Perform k-NN and MICE imputation
impyute	Perform EM imputation
sklearn.preprocessing.Imputer	Perform Mean imputation
sklearn.model_selection.KFold	Perform K-Fold Cross Validation
imblearn.over_sampling.SMOTE	Perform SMOTE Oversampling
xgboost.XGBClassifier	Extreme Gradient Boosting classifier
sklearn.ensemble.RandomForestClassifier	Random Forest Classifier
sklearn.linear_model.LogisticRegression	Logistic Regression Classifier
imblearn.ensemble.BalancedBaggingClassifier	Balanced Bagging Classifier
sklearn.tree.DecisionTreeClassifier	Decision Tree Classifier
sklearn.naive_bayes.GaussianNB	Gaussian Naïve Bayes Classifier
sklearn.metrics	Performance evaluation metrics like accuracy score, recall, precision, ROC curve, etc.

Table 5: Libraries used for the project.

1. Firstly, we imported all the libraries we listed in Table 5.
2. Then we load the raw data (.arff files) as pandas dataframes and assign the new column headers to them. Although the features are numeric and class labels are binary, in the dataframes, all the values were stored as objects. So we converted them to float and int values respectively.
3. Now we start the data analysis. Firstly, we see how much of data is missing in each dataframe and look at the nullity (sparsity) by generating the nullity matrix and nullity heatmaps respectively.

4. Then we perform imputation of the missing data using Mean, k-NN, EM and MICE imputation techniques and generate fresh dataframes of imputed data.
5. We apply SMOTE oversampling on all these imputed dataframes to get fresh dataframes of imputed-and-oversampled dataframes and store them in a dictionary.
6. We create (instantiate) the 6 classifier models (GNB, LR, DT, RF, XGB, BB) and store them in a dictionary.
7. We iterate over all the models. In each model, we iterate over all the 4 imputed-oversampled dataset collections. Each collection has 5 dataframes corresponding to 5 years' data. On each of these years' datasets, we train the model using K-Folds Cross Validation and store the results in nested dictionaries.
8. We use the nested dictionaries of results to export the results as CSV files and generate charts using Excel.

4. Results

Our results are organized as follow: Firstly, we report the accuracy score of the 6 models we have experimented with, using a plot of the accuracy score against each of the imputation method (Mean, k-NN, EM and MICE), and internally, on each of the 5 datasets (Year 1 – Year 5). Later, we also report the accuracy scores by years' datasets, i.e., plot of accuracy scores for each year's dataset, plotted against the 4 imputation techniques, and internally, the 6 models.

4.1 Accuracy plots of models

4.1.1 Gaussian Naïve Bayes (GNB)

The accuracy plot of Gaussian Naïve Bayes classifier model is shown in Figure 4. It was not surprising to us that the performance of GNB model was poor across all the imputation techniques. The overall accuracy scores remained below 55% and it only slightly better than a naïve random guess with a probability of 0.5. The highest accuracy was obtained for 3rd year dataset under the MICE imputation. However, if we averaged the accuracy across all the years (1-5), the highest accuracy of 51.77% was output by EM imputation.

4.1.2 Logistic Regression (LR)

The accuracy plot of Logistic Regression classifier model is shown in Figure 5. This experiment was conducted using L1 regularization and the regularization parameter used was 1. We expected LR to perform better than GNB classifier model but it was quite surprising to see the overall (averaged) accuracy score of LR lower than GNB.

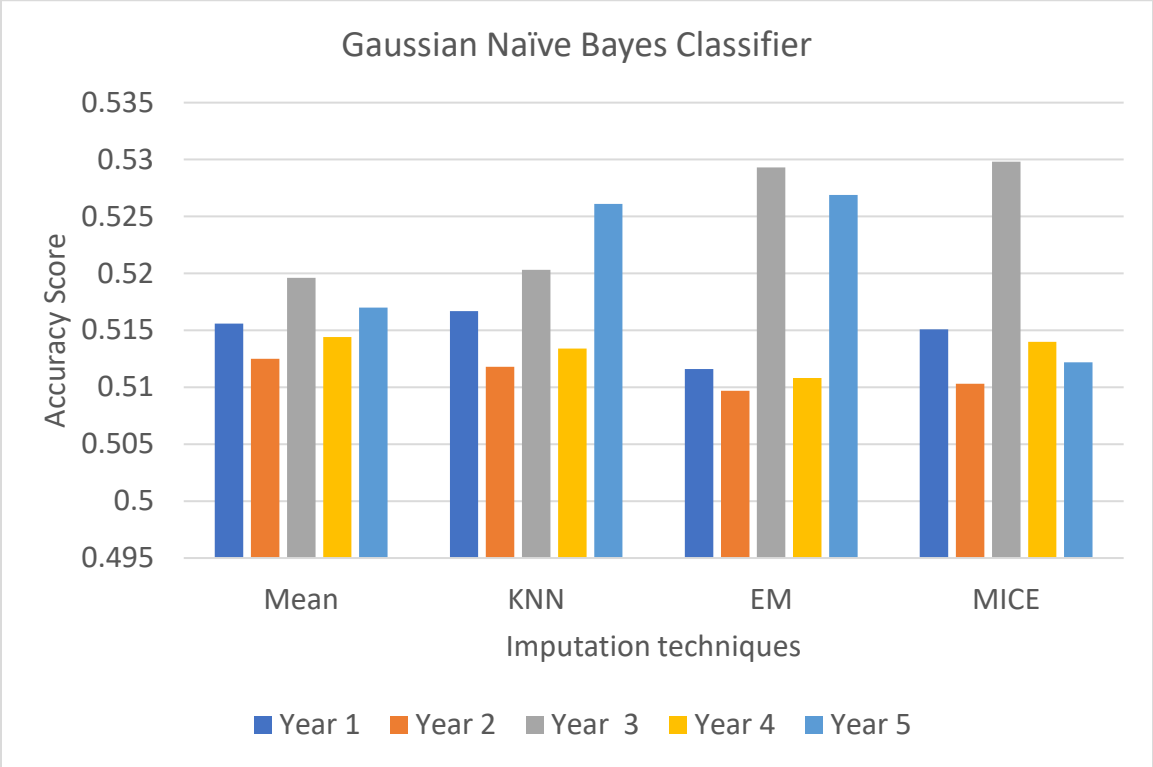

Figure 4: Accuracy evaluation of Gaussian Naïve Bayes classifier.

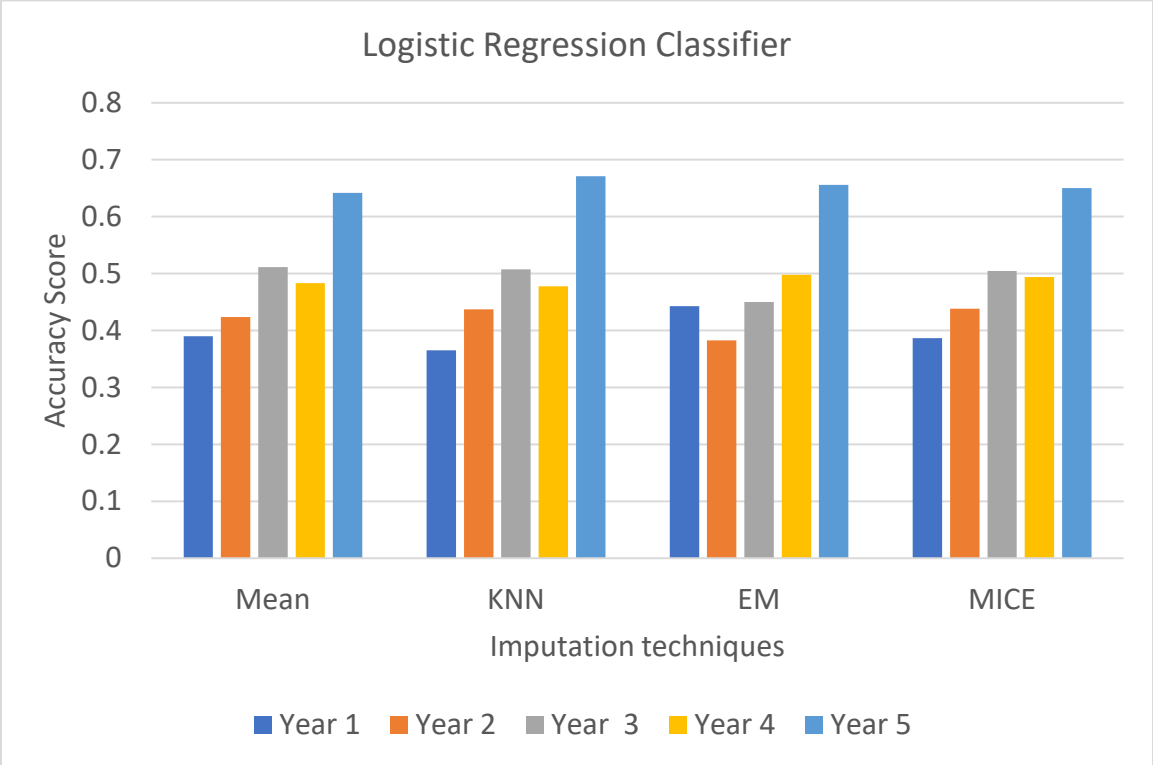

Figure 5: Accuracy evaluation of Logistic Regression classifier.

Although the Year 5 dataset for all the imputation techniques obtained an accuracy of $>$ 60%, the average accuracy of all the years is lower than the GNB, when compared with the corresponding imputation techniques.

4.1.3 Decision Tree (DT)

The accuracy plot of Decision Tree Classifier model is shown in Figure 6. Decision Tree classifier model performed slightly better than what we expected. It performed much better than the LR and GNB models we saw previously. The highest accuracy of 93.58% was obtained by the dataset Year 1 for Mean imputation technique. The highest average accuracy was also for the Mean imputed datasets. All the datasets under k-NN imputation performed poorly somehow, with an average accuracy of only 87.48%. Year 4 datasets consistently performed poorly for all the imputation methods.

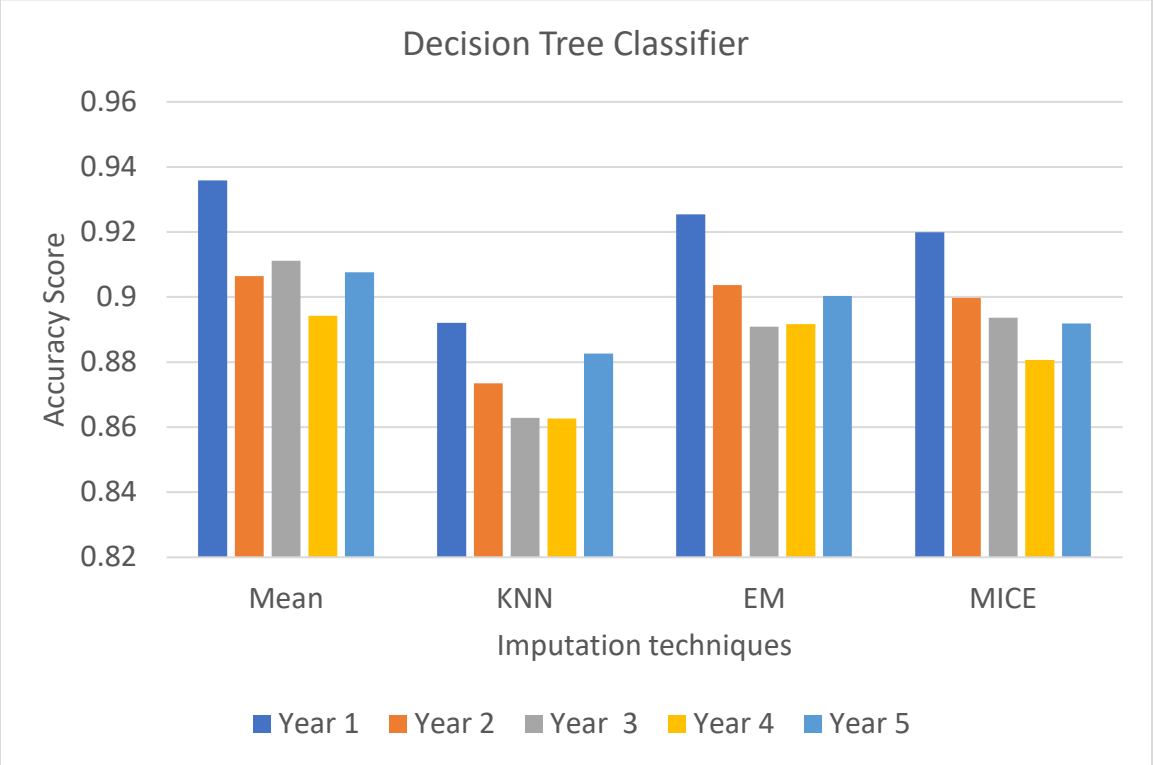

Figure 6: Accuracy evaluation of Decision Tree classifier.

4.1.4 Random Forests (RF)

The accuracy plot of Random Forest Classifier model is shown in Figure 7. As we expected, RF model performed better than the Decision Tree model. RF model obtained its highest accuracy of 92.89% for Mean imputation method, where Decision Tree model

only could obtain 91.10%. The highest difference between the performance of RF (90.28%) and DT (87.48%) models was noted for the k-NN imputation method.

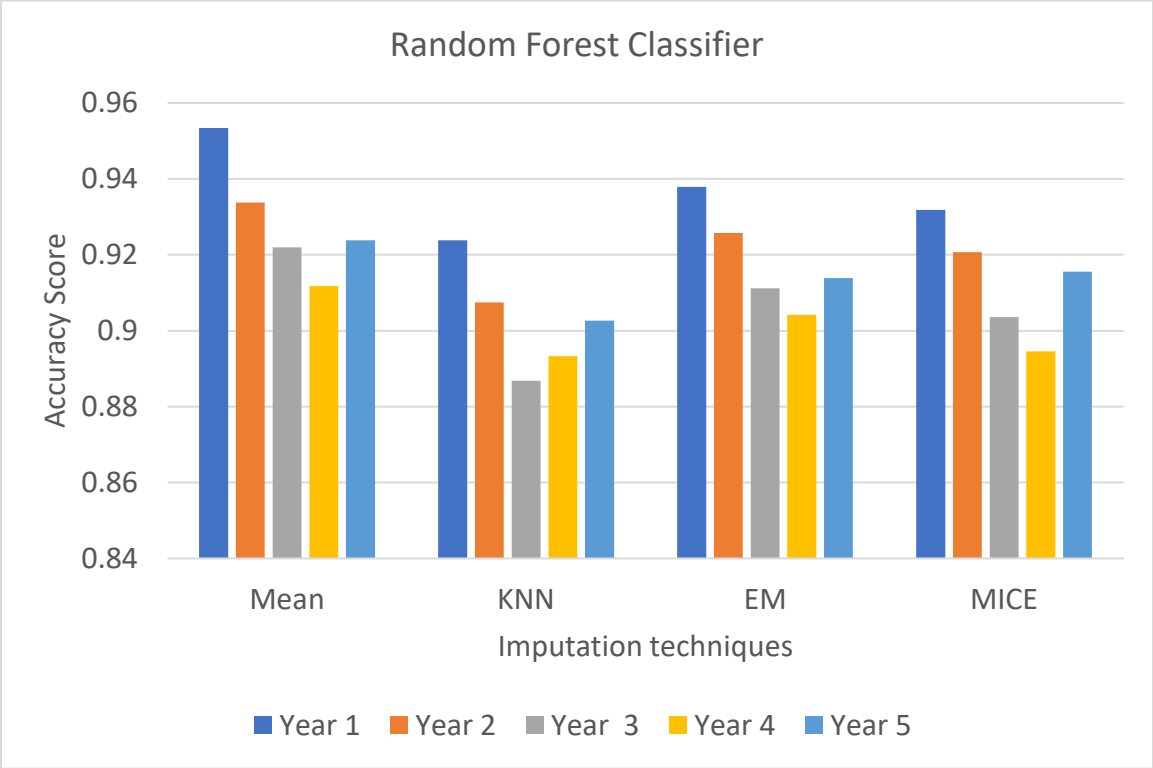

Figure 7: Accuracy evaluation of Random Forest classifier.

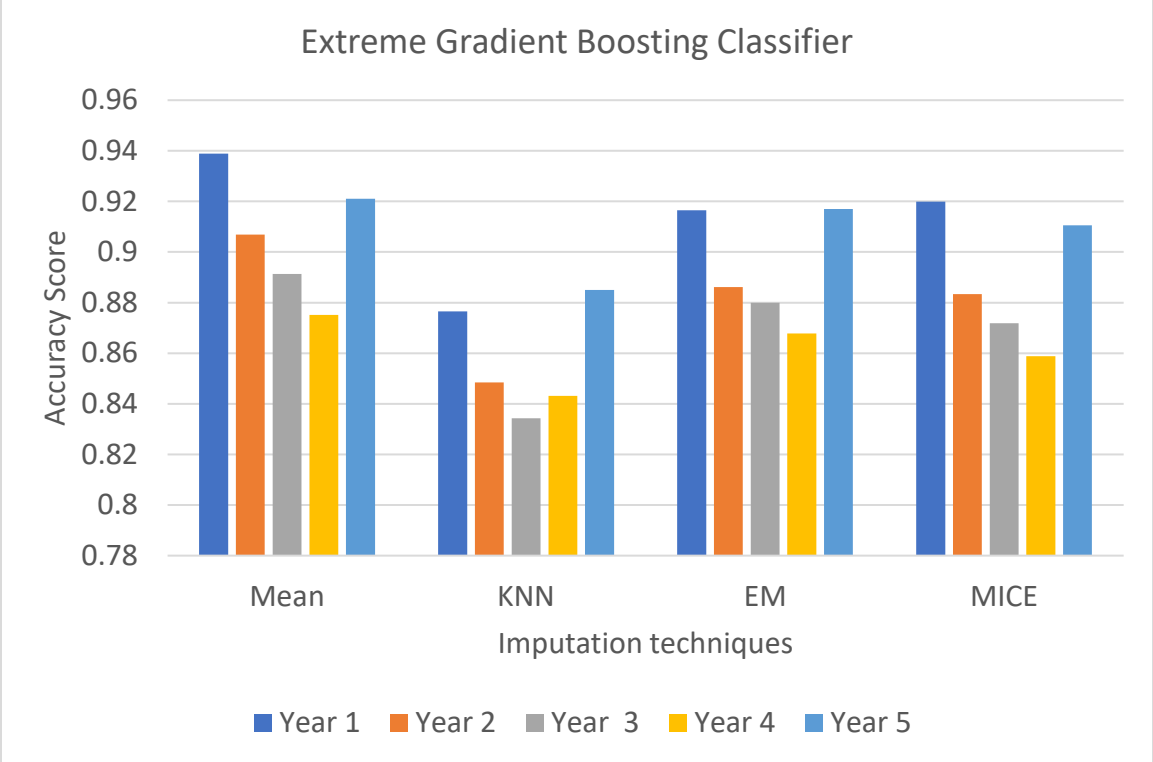

Figure 8: Accuracy evaluation of Extreme Gradient Boosting classifier.

4.1.5 Extreme Gradient Boosting (XGB)

The accuracy plot of Extreme Gradient Boosting Classifier model is shown in Figure 8 above. We expected XGB model would perform better than the Decision Tree and Random Forest methods, but it performed slightly worse than the Decision Tree model. The highest accuracy was obtained by the XGB model was for the Year 1 dataset and the Mean imputation method. All the datasets performed poorly for the k-NN imputation method.

4.1.6 Balanced Bagging (BB)

The accuracy plot of Balanced Bagging Classifier model is shown in Figure 9 below. Balanced Bagging was the best model we have experimented with so far. It has given the highest accuracy scores which were higher than all other models for each of the data imputation methods. The highest accuracy of 98.19% was obtained for the Year 1 dataset and Mean imputation method. The least accuracy of 94.2% was obtained for Year 5 dataset for k-NN imputation. Even this least accuracy score was better than the accuracy scores reported by the other models. The highest average accuracy across all years' datasets of 96.59% was obtained for the mean imputation method.

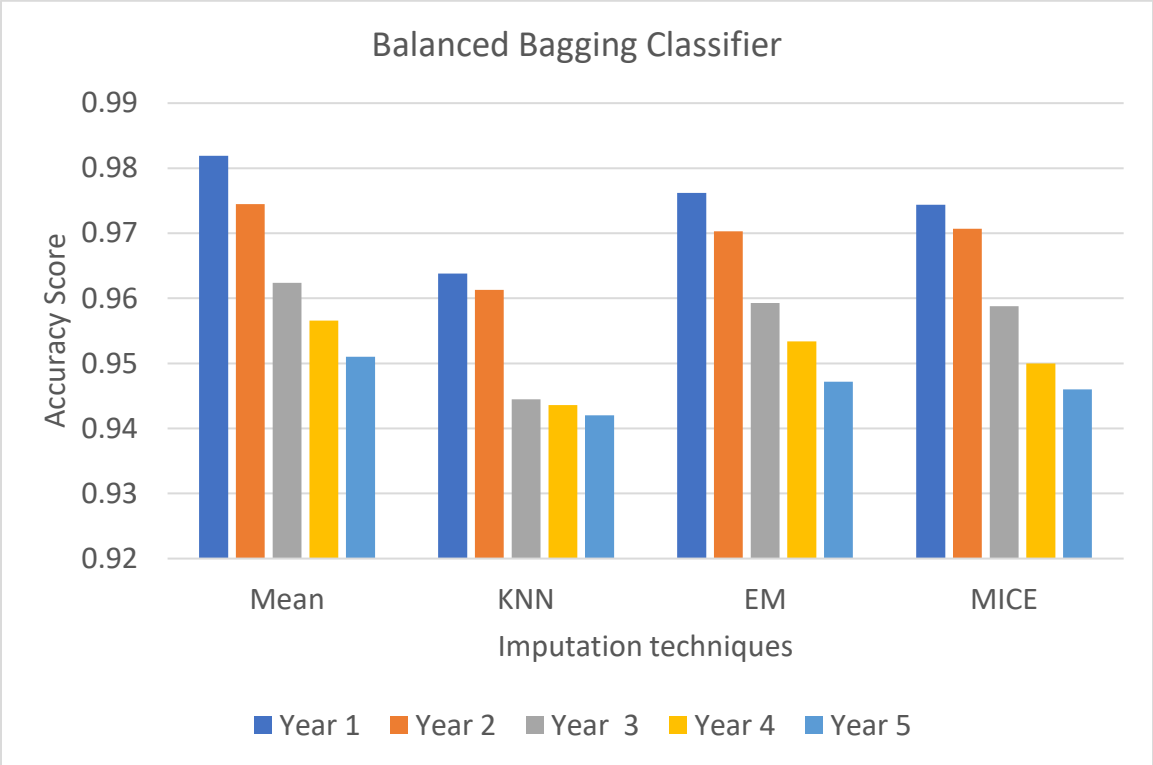

Figure 9: Accuracy evaluation of Balanced Bagging classifier.

4.2 Accuracy plots by Years' datasets

4.2.1 Year 1 dataset

The accuracy plot for the Year 1 dataset is shown in Figure 10. We observed that for Year 1, Balanced Bagging classifier model with Mean imputation methods proven to be the most successful once with an accuracy of 98.19%. In general, more or less, all the imputation techniques performed uniformly. The next best combination would be Random Forest model with Mean Imputation with an accuracy of 95.37%.

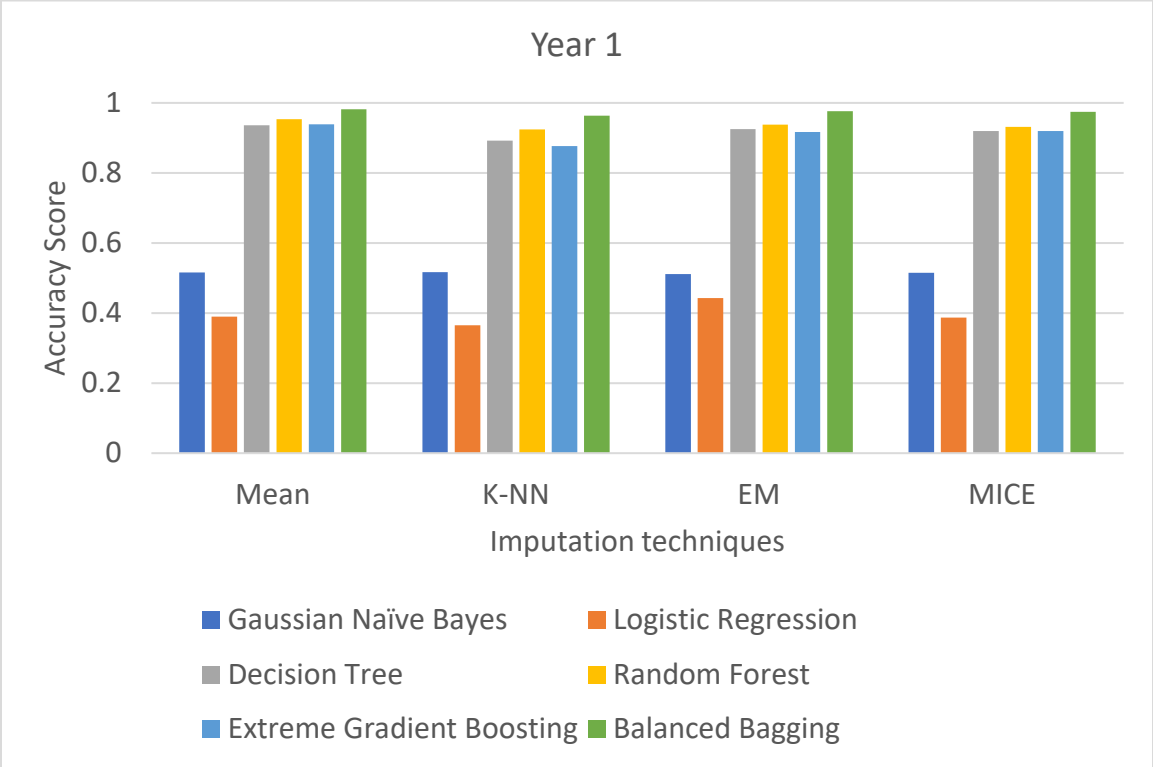

Figure 10: Accuracy evaluation of Year 1 dataset

4.2.2 Year 2 dataset

The accuracy plot for the Year 2 dataset is shown in Figure 11. Like the Year 1 dataset, Year 2 dataset also obtained its best performance for Balanced Bagging classifier model and Mean Imputation method with an accuracy of 97.45%. The next best model is Random Forest with Mean imputation method with an accuracy of 93.38%

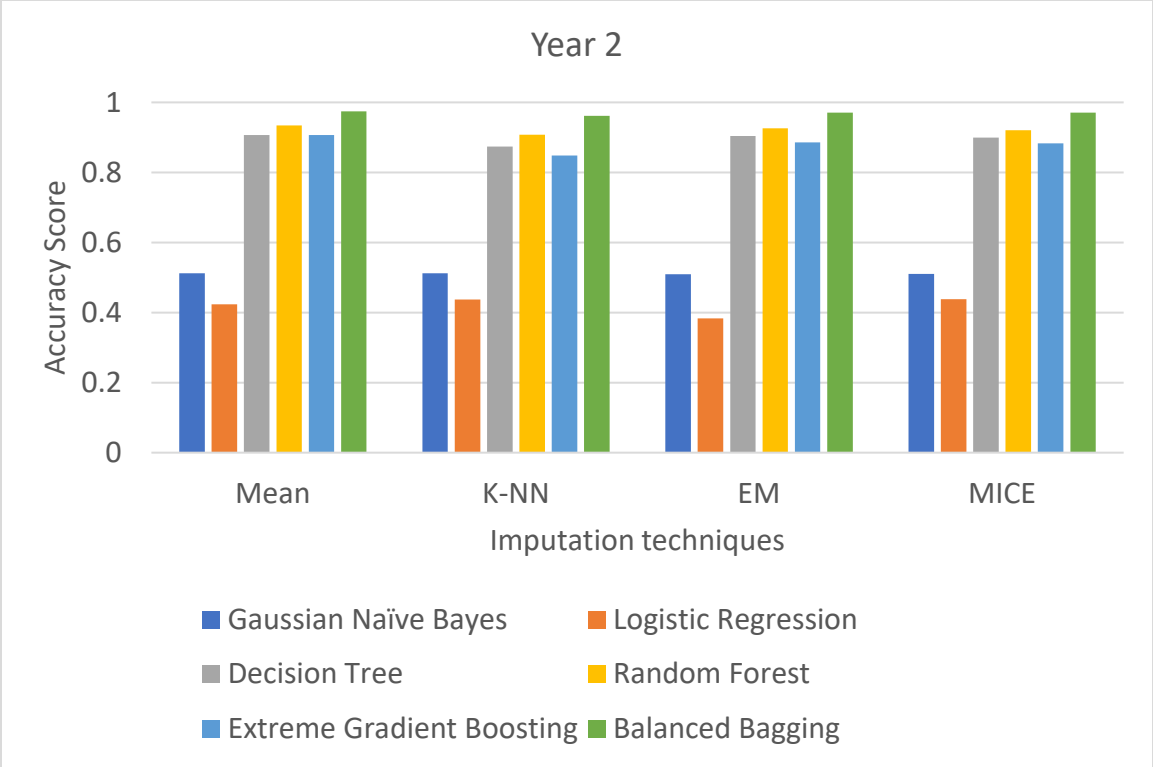

Figure 11: Accuracy evaluation of Year 2 dataset.

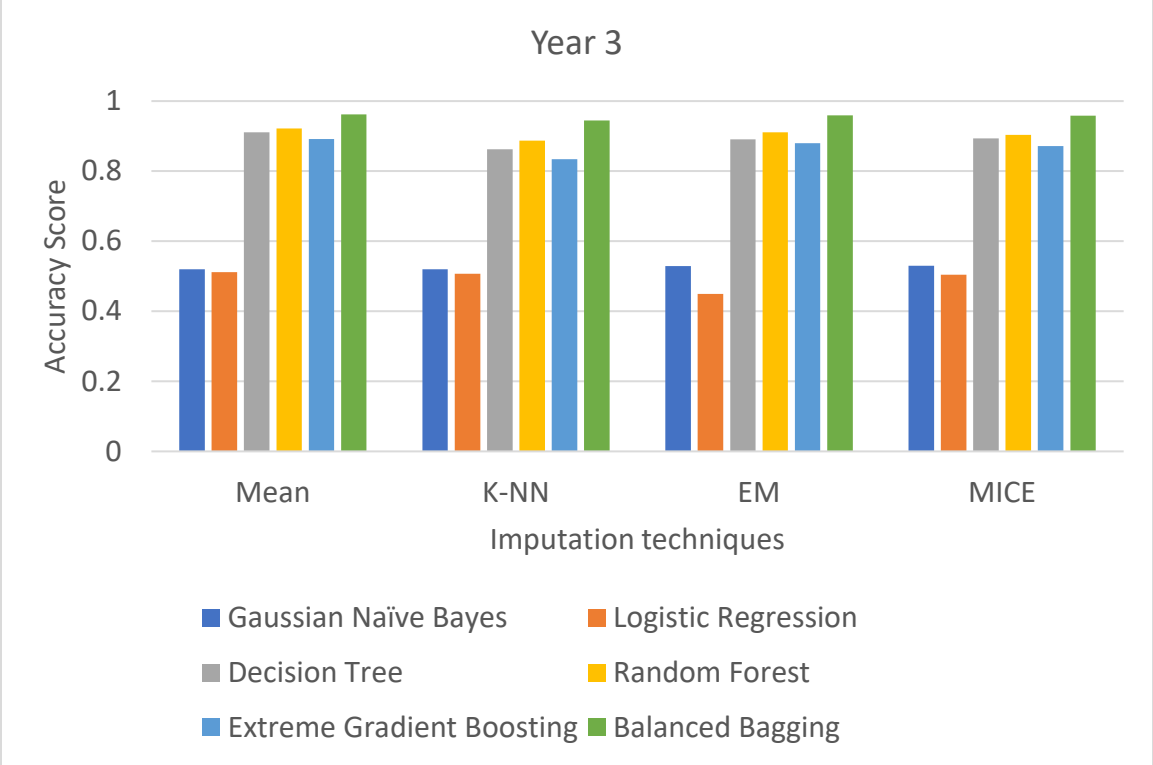

Figure 12: Accuracy evaluation of Year 3 dataset.

4.2.3 Year 3 dataset

The accuracy plot for the Year 3 dataset is shown in Figure 12. Like the Year 1 and Year 2 datasets, Year 3 dataset also obtained its best performance for Balanced Bagging classifier model and Mean Imputation method with an accuracy of 96.24%, closely followed by the EM imputation for the same model, with an accuracy of 95.93%. The next best model is Random Forest with Mean imputation method with an accuracy of 92.22%

4.2.4 Year 4 dataset

The accuracy plot for the Year 4 dataset is shown in Figure 13 below. Like the previous plots, Year 4 dataset also obtained its best performance for Balanced Bagging classifier model and Mean Imputation method with an accuracy of 95.66%, closely followed by the EM imputation for the same model, with an accuracy of 95.34%. The next best model is Random Forest with Mean imputation method with an accuracy of 91.18%.

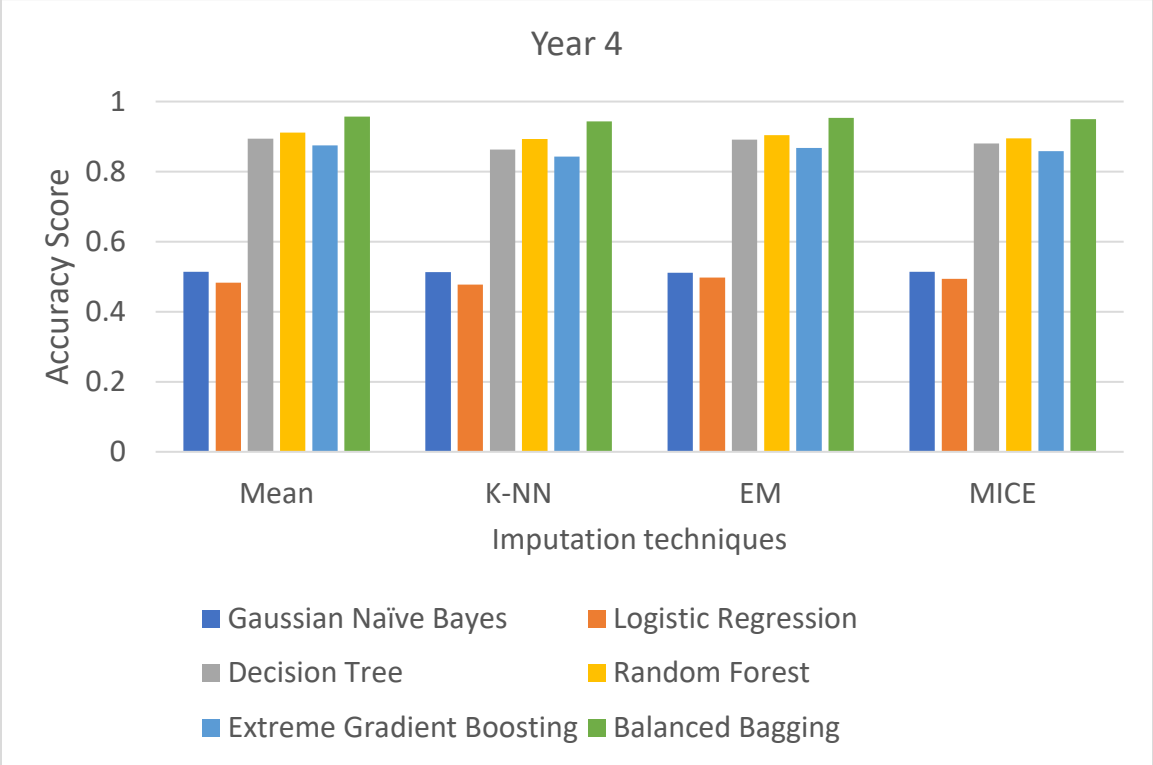

Figure 13: Accuracy evaluation of Year 4 dataset.

4.2.5 Year 5 dataset

The accuracy plot for the Year 5 dataset is shown in Figure 14 below. Like the previous plots, Year 5 dataset also obtained it's best performance for Balanced Bagging classifier model and Mean Imputation method with an accuracy of 95.10%, closely followed by the EM imputation for the same model, with an accuracy of 94.72%. The next best model is Extreme Gradient Boosting with Mean imputation method with an accuracy of 92.10%.

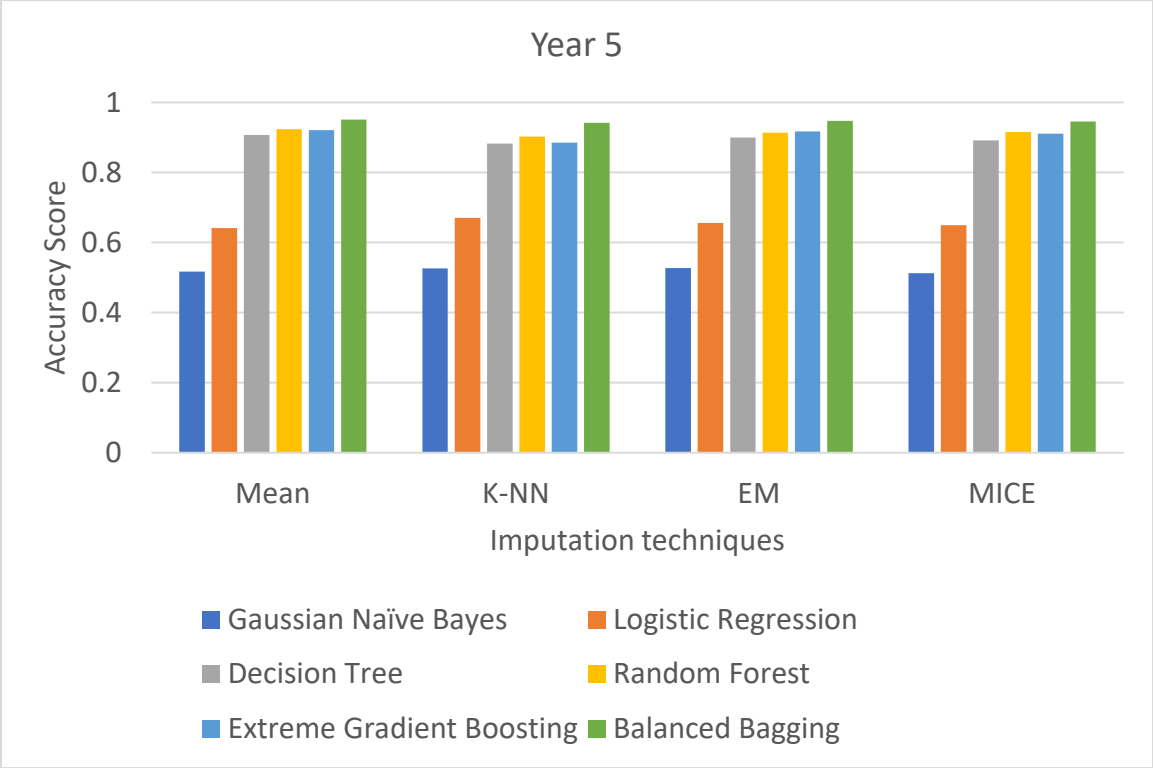

Figure 14: Accuracy evaluation of Year 5 dataset.

		IMPUTATION TECHNIQUES			
		Mean	**k-NN**	**EM**	**MICE**
MODELS	**Gaussian Naïve Bayes**	51.58	51.77	51.77	51.63
	Logistic Regression	49.00	49.16	48.58	49.48
	Decision Tree	91.10	87.48	90.24	89.72
	Random Forests	92.89	90.28	91.86	91.32
	Extreme Gradient Boosting	90.67	85.75	89.36	88.82
	Balanced Bagging	96.59	95.14	96.07	95.86

Table 6: Mean accuracies across all years' datasets for various models and imputation methods

5. Discussion

As we have seen the commentary in results section, the best model we have experimented with so far is the Balanced Bagging Classifier, with the Mean Imputation method. We have implemented the Balanced Bagging Classifier with Decision Tree as a base estimator and n-estimators as > 5. The number of samples to draw from the give training data X to train each base estimator is 1. The number of features to draw from the training set X to train each base estimator is also 1. The samples are drawn with replacement, and are not using the out-of-bag samples to estimate the generalization error. Figure 15 shows the effect of varying the number of estimators in the Balanced Bagging classifier, for various years' datasets. We could observe from the figure that as the number of estimators grew, initially, the accuracy plots peaked up quickly. After some estimators (10), the accuracy plots began to converge, although they pulsed slightly. The highest accuracy was obtained by the Year 1 dataset and the least accuracy was obtained by the Year 5 dataset. This was also depicted in Figure 16. It shows the accuracy performance of Balanced Bagging classifier by years' datasets, for the Mean imputation technique.

Table 6 summarizes the mean accuracies across all the years' datasets for various models and imputation methods. Looking, at the performance of the models on each imputation technique, we were surprised by the statistics of the mean imputation technique. We were expecting that the MICE imputation technique will account for the best accuracy of models, and the mean imputation technique was a baseline imputation method. But it turned out that Mean imputation gave better results than most other imputation techniques, even though the logic of its operation is naïve and simple.

When it comes to model ranking, Balanced Bagging outperforms all other models in terms of accuracy, for all the imputation techniques. While Extreme Gradient Boosting was expected to show up as the next best model, it ended up in the third place, giving away the second place to Random Forest model. Random Forest model, as we seen in its analysis in the results section, performed slightly better than the Decision Tree model, thus pushing the Decision Tree model, to the fourth place. Although Logistic Regression was expected to perform better than the Gaussian Naïve Bayes model, it showed the worst performance among all the models and the Gaussian Naïve Bayes model was only slightly better than the Logistic Regression model, and ranked 5^{th} best model.

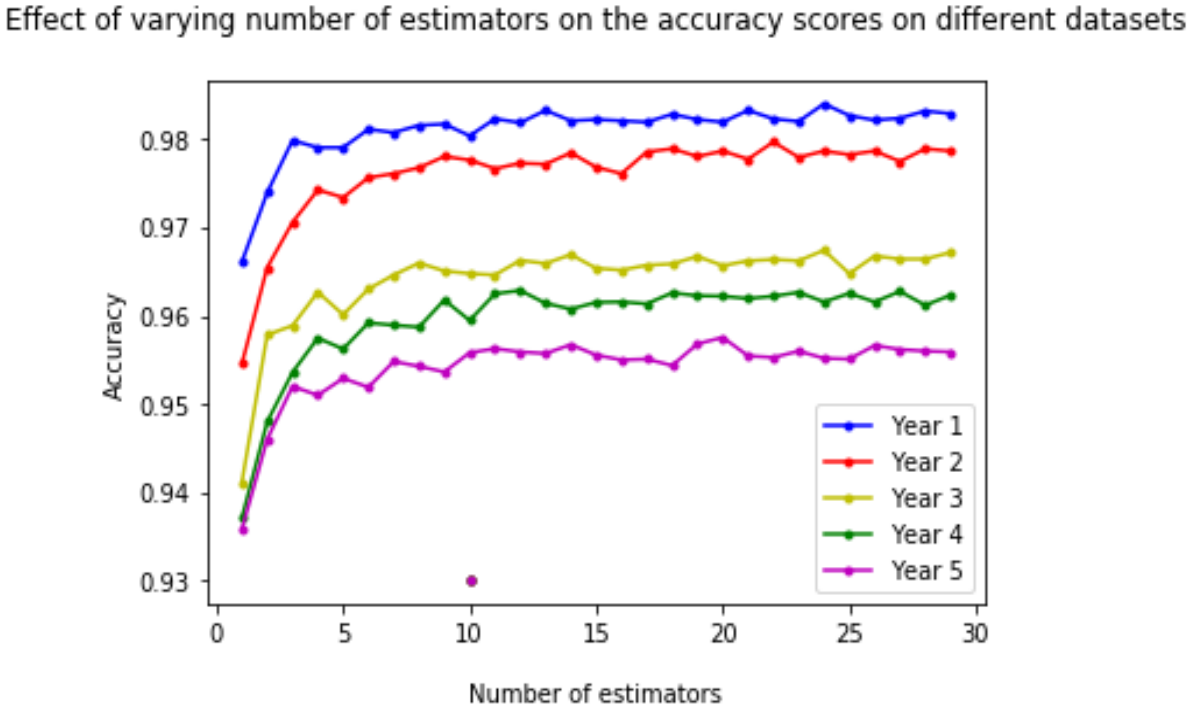

Figure 15: Effect of varying the estimators in Balanced Bagging Classifier

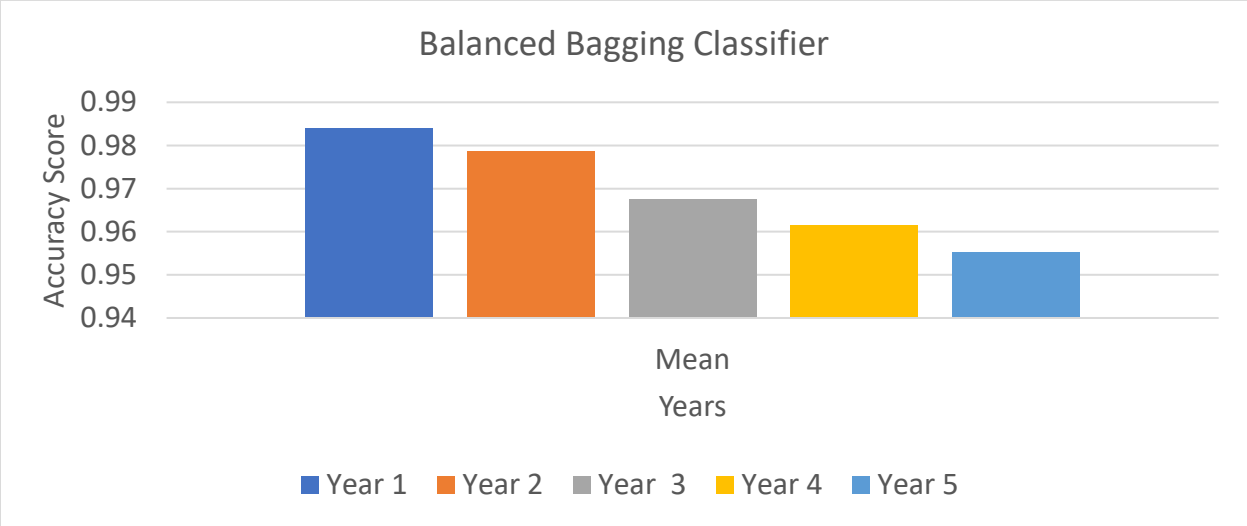

Figure 16: Performance comparison of various years' data for Balanced Bagging model.

6. Future Work

In this section, we intend to discuss the future work in bankruptcy prediction. So far, in our experiments, we have dealt with the synthetic features—an arithmetic combination of core econometric features. But it is also possible to gather more core features and hence synthesize more synthetic features by varying the arithmetic operations performed on these core features. Also, it is possible to synthesize more features considering the current synthetic features as base features. Doing so may result in better prediction of bankruptcy, but it has to be thoroughly validated by domain experts, as to whether such highly complex synthetic features would be meaningful, in terms of financial economics. It is also feasible to reduce the dimensionality of features. But for a dataset like the Polish bankruptcy data we have seen so far, with such high missing data, it becomes difficult to rank the features and perform feature extraction. The features dropped in such manner might actually bear significant impact in the prediction, had the data not been so much sparse. So, the takeaway is that, if the data to be collected in the future, pertinent to bankruptcy prediction, is made sure to less sparse, it is possible to apply all the techniques mentioned in the future scope so far, and hence obtain better predictive models.

7. Conclusion

In this section we discuss the summary of our work done in this project thus far. We have successfully modelled 6 classification models: Gaussian Naïve Bayes, Logistic Regression, Decision Trees, Random Forests, Extreme Gradient Boosting and Balanced Bagging classifiers. The training sets were made sure to have a balanced sets of class labels, by oversampling the minority class labels using Synthetic Minority Oversampling technique. Also, we have imputed the missing values in the data using 4 imputer techniques: Mean, k-Nearest Neighbors (k-NN), Expectation-Maximization (EM) and Multiple Imputation using Chained Equations (MICE). The biggest challenge was to deal with the missing/sparse data. Since all the companies being evaluated for bankruptcy don't operate on the same timelines, it is difficult to gather meaningful data and organize it. The features on which the bankruptcy prediction is based, are not as straightforward as the financial ratios found on the balance sheets of the companies, and need to thoroughly be studied and validated. We have successfully documented our findings and suggested the best bankruptcy prediction model we have seen in our project.

8. References

[1] Zieba, M., Tomczak, S., Tomczak, J. (2016). Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. Expert Systems with Applications, 2016.

[2] Zhang, Y., Wang, S., & Ji, G. (2013). A rule-based model for bankruptcy prediction based on an improved genetic ant colony algorithm. Mathematical Problems in Engineering, 2013.

[3] Wikipedia contributors. "Bankruptcy prediction". Wikipedia, The Free Encyclopedia, <https://en.wikipedia.org/wiki/Bankruptcy_prediction>

[4] Constand, R. L., & Yazdipour, R. (2011). Firm failure prediction models: a critique and a review of recent developments. Advances in Entrepreneurial Finance (pp. 185–204). Springer.

[5] Altman, E. I., & Hotchkiss, E. (2010). Corporate financial distress and bankruptcy: Predict and avoid bankruptcy, analyze and invest in distressed debt: 289. Hoboken, New Jersey: John Wiley & Sons.

[6] Koh, H. C., & Killough, L. N. (1990). The use of multiple discriminant analysis in the assessment of the going-concern status of an audit client. Journal of Business Finance & Accounting, 17, 179–192.

[7] Laitinen, E. K. (1991). Financial ratios and different failure processes. Journal of Business Finance & Accounting, 18, 649–673.

[8] Wilcox, J. W. (1973). A prediction of business failure using accounting data. Journal of Accounting Research, 11, 163–179.

[9] Chen, T., & He, T. (2015b). xgboost: extreme gradient boosting. R package version 0.3-0. Technical Report .

[10] Friedman JH (2001). "Greedy function approximation: a gradient boosting machine." Annals of Statistics, pp. 1189–1232.

[11] Bache K, Lichman M (2013). "UCI Machine Learning Repository." URL http://archive.ics. uci.edu/ml. Friedman J, Hastie T, Tibshirani R, et al. (2000). "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)." The annals of statistics, 28(2), 337–407. Friedman JH (2001). "Greedy function approximation: a gradient boosting machine." Annals of Statistics, pp. 1189–1232.

[12] Friedman J, Hastie T, Tibshirani R, et al. (2000). "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)." The annals of statistics, 28(2), 337–407.

[13] Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998). On combining classifiers. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20, 226–239.

[14] Quinlan, J.R. (1983a). Learning efficient classification procedures and their application to chess endgames. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell, (Eds.), Machine learning: An artificial intelligence approach. Palo Alto: Tioga Publishing Company.

[15] Merwin, C. L. (1942). Financing small corporations in five manufacturing industries. NBER Books p. 1926-1936. New York: National Bureau of Economic Research, Inc.

[16] Shapiro, A. (1983). The role of structured induction in expert systems. Ph.D. Thesis, University of Edinburgh.

[17] Sinkey, J. F. (1975). A multivariate statistical analysis of the characteristics of problem banks. The Journal of Finance, 30, 21–36.